**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**по лабораторной работе №4**

**по дисциплине «Программирование»**

**Тема: Динамические структуры данных**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 3342 |  | Смирнова Е.С. |
| Преподаватель |  | Глазунов С.А. |

Санкт-Петербург

2024

## **Цель работы**

Цель работы заключается в изучении способов реализации динамических структур данных на языке C++ и их применении для решения конкретной задачи. В рамках данной работы необходимо написать программу, которая использует одну из динамических структур данных — стек, для решения задачи валидации тегов html документа.

## **Задание**

Вариант 6.

Расстановка тегов.

Требуется написать программу, получающую на вход строку, (без кириллических символов и не более 3000 символов) представляющую собой код "простой" html-страницы и проверяющую ее на валидность. Программа должна вывести correct если страница валидна или wrong.

html-страница, состоит из тегов и их содержимого, заключенного в эти теги. Теги представляют собой некоторые ключевые слова, заданные в треугольных скобках. Например, <tag> (где tag - имя тега). Область действия данного тега распространяется до соответствующего закрывающего тега </tag> который отличается символом /. Теги могут иметь вложенный характер, но не могут пересекаться.

<tag1><tag2></tag2></tag1> - верно

<tag1><tag2></tag1></tag2> - не верно

Существуют теги, не требующие закрывающего тега.

Валидной является html-страница, в коде которой всякому открывающему тегу соответствует закрывающий (за исключением тегов, которым закрывающий тег не требуется).

Во входной строке могут встречаться любые парные теги, но гарантируется, что в тексте, кроме обозначения тегов, символы < и > не встречаются, атрибутов у тегов также нет.

Теги, которые не требуют закрывающего тега: <br>, <hr>.

Стек (который потребуется для алгоритма проверки парности тегов) требуется реализовать самостоятельно на базе массива. Для этого необходимо:

Реализовать класс CustomStack, который будет содержать перечисленные ниже методы. Стек должен иметь возможность хранить и работать с типом данных char\*

Перечень методов класса стека, которые должны быть реализованы:

void push(const char\* val) - добавляет новый элемент в стек

void pop() - удаляет из стека последний элемент

char\* top() - доступ к верхнему элементу

size\_t size() - возвращает количество элементов в стеке

bool empty() - проверяет отсутствие элементов в стеке

extend(int n) - расширяет исходный массив на n ячеек

## **Выполнение работы**

В рамках выполнения работы была разработана программа на языке C++, включающая в себя класс CustomStack.

Экземпляры класса содержат защищенное поле mData для хранения массива отдельных тегов и приватные поля mIndex для обращения к верхнему элементу стека и размер текущей памяти, зарезервированной под стек. Класс CustomStack реализует структуру данных стек на основе массива.

Класс предоставляет следующие методы:

• push: добавляет новый элемент на верх стека;

• top: возвращает значение верхнего элемента;

• pop: вытаскивает значение верхнего элемента, удаляя его из стека;

• empty: возвращает true, если стек пуст, иначе false;

• size: возвращает текущее количество элементов в стеке;

• extend: расширяет память, выделенную под стек, на указанное

количество элементов.

Разработанный программный код см. в приложении А.

## **Тестирование**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
| 1. | <tag1><tag2></tag2></tag1> | correct |
| 2. | <tag1><tag2></tag1></tag2> | wrong |
| 3. | <html><head><title>HTML  Document</title></head><bo  dy><p><b>This text is  bold,<br><i>this  is bold and  italics</i></b></p></body>  </html> | correct |

## **Выводы**

В результате выполнения данной работы были изучены принципы реализации динамических структур данных на языке C++ и их применение для решения задачи валидации тегов html документа. Была разработана программа, использующая стек в качестве структуры данных для проверки корректности расстановки html-тегов.

# **ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: main.c

class CustomStack {

public:

// Constructor

CustomStack(){

this->mData = new char \*[this->mCapacity];

if (!(this->mData)) {

cerr << "Error" << endl;

exit(0);

}

this->mIndex = -1;

}

// Destructor

~CustomStack(){

for (int i = 0; i <= mIndex; i++){

delete mData[i];

}

delete[] mData;

}

void push(const char \*val){

this->mIndex++;

if (this->mIndex >= this->mCapacity){

extend(this->mCapacity);

}

this->mData[this->mIndex] = new char[strlen(val) + 1];

if (!(this->mData[this->mIndex])) {

cerr << "Error" << endl;

exit(0);

}

strcpy(this->mData[this->mIndex], val);

}

char \*top(){

return this->mData[this->mIndex];

}

char \*pop(){

return this->mData[this->mIndex--];

}

bool empty(){

return this->mIndex == -1;

}

size\_t size(){

return this->mIndex + 1;

}

void extend(int n){

char \*\*newData = new char \*[this->mCapacity + n];

if (newData == nullptr) {

cerr << "Error" << endl;

exit(0);

}

for (size\_t i = 0; i <= this->mIndex; ++i){

newData[i] = this->mData[i];

}

delete[] mData;

this->mCapacity += n;

this->mData = newData;

}

private:

int mIndex;

size\_t mCapacity = 10;

protected:

char \*\*mData;

};

int main(){

string commands;

getline(cin, commands);

CustomStack st;

for (int i = 0; i < commands.size(); i++) {

char tag[10];

if (commands[i] == '<') {

int j = i + 1, n = 0;

while (commands[j] != '>') {

tag[n] = commands[j];

n++;

j++;

}

tag[n] = '\0';

if (tag[0] == '/') {

if (st.empty()){

cout << "wrong";

return 0;

}

char \*check = st.top();

for (int k = 1; tag[k]; k++){

if (check[k - 1] != tag[k]){

cout << "wrong";

return 0;

}

}

st.pop();

}

else if(strcmp(tag, "br") != 0 && strcmp(tag, "hr") != 0){

st.push(tag);

}

i = j;

}

}

if (st.empty()){

cout << "correct";

return 0;

}

cout<<"wrong";

return 0;

}